AMENDMENTS TO THE SPECIFICATION:

Please amend the paragraph beginning at page 4, line 14, as follows:

The invention simplifies the provision of scheduling support by providing a system in which program instructions are sent to the hardware based instruction execution unit and forwarded from there to the software based instruction execution unit if they cannot be ~~deal~~ dealt with by the hardware based instruction execution unit. In this way, by routing all the program instructions through the hardware based instruction execution unit, this unit is able to keep track of the execution of instructions and accordingly generate a scheduling signal for triggering a scheduling operation irrespective of whether the preceding instructions have been executed by hardware or software.

Please amend the paragraph beginning at page 10, line 22, as follows:

The set of registers may be empty, partly filled with stack operands or completely filled with stack operands. The particular register that currently holds the top of stack operand may be any of the registers within the set of registers. It will thus be appreciated that the instruction translator may be in any one of seventeen different mapping states corresponding to one state when all of the registers are empty and four groups of four states each corresponding to a respective different number of stack operands being held within the set of registers and with a different register holding the top of stack operand. Table 1 illustrates the seventeen different states of the state mapping for the instruction translator 108. It will be appreciated that with a different number of registers allocated

for stack operand storage, or as a result of constraints that a particular processor core may have in the way it can manipulate data values held within registers, the mapping states can ~~very~~ vary considerably depending upon the particular implementation and Table 1 is only given as an example of one particular implementation.

Please amend the paragraph beginning at page 11, line 39, as follows:

Within Table 1 it may be observed that the first three bits of the state value indicate the number of non-empty registers within the set of registers. The final two bits of the state value indicate the register number of the register holding the top of stack operand. In this way, the state value may be readily used to control the operation of a hardware translator or a software translator to take account of the ~~currently~~ current occupancy of the set of registers and the current position of the top of stack operand.

Please amend the paragraph beginning at page 26, line 30, as follows:

Figure 9 illustrates a Java bytecode translation unit ~~68~~ 64 that receives a stream of Java bytecodes and outputs a translated stream of ARM instructions (or corresponding control signals) to control the action of a processor core. As described previously, the Java bytecode translator ~~68~~ 64 translates simple Java bytecodes using instruction templates into ARM instructions or sequences of ARM instructions. When each Java bytecode has been executed, then a counter value within scheduling control logic 70 is decremented. When this counter value reaches 0, then the Java bytecode translation unit ~~68~~ 64 issues an ARM instruction branching to scheduling code that manages scheduling between threads or tasks as appropriate.

Please amend the paragraph beginning at page 27, line 5, as follows:

Whilst simple Java bytecodes are handled by the Java bytecode translation unit ~~68~~ 64 itself providing high speed hardware based execution of these bytecodes, bytecodes requiring more complex processing operations are sent to a software interpreter provided in the form of a collection of interpretation routines (examples of a selection of such routines are given earlier in this description). More specifically, the Java bytecode translation unit ~~68~~ 64 can ~~determined~~ determine that the bytecode it has received is not one which is supported by hardware translation and accordingly a branch can be made to an address dependent upon that Java bytecode where a software routine for interpreting that bytecode is found or referenced. This mechanism can also be employed when the scheduling logic 70 indicates that a scheduling operation is needed to yield a branch to the scheduling code.